T1B2 1555
Proceedings of the 2005 IEEE
Workshop on Information Assurance and Security
United States Military Academy, West Point, NY, 15–17 June 2005

# Towards Collaborative Security and P2P Intrusion Detection

Michael E. Locasto, Janak J. Parekh, Angelos D. Keromytis, Salvatore J. Stolfo

*Abstract*— **The increasing array of Internet–scale threats is a pressing problem for every organization that utilizes the network. Organizations have limited resources to detect and respond to these threats. The end-to-end (E2E) sharing of information related to probes and attacks is a facet of an emerging trend toward "collaborative security."**

**The key benefit of a collaborative approach to intrusion detection is a better view of global network attack activity. Augmenting the information obtained at a single site with information gathered from across the network can provide a more precise model of an attacker's behavior and intent. While many organizations see value in adopting such a collaborative approach, some challenges must be addressed before intrusion detection can be performed on an inter–organizational scale.**

**We report on our experience developing and deploying a decentralized system for efficiently distributing alerts to collaborating peers. Our system, *Worminator*, extracts relevant information from alert streams and encodes it in Bloom Filters. This information forms the basis of a distributed watchlist. The watchlist can be distributed via a choice of mechanisms ranging from a centralized trusted third party to a decentralized P2P-style overlay network.**

## I. Introduction

As the threat to critical infrastructure computer networks grows, the ability to rapidly and correctly identify, rank, and react to probes and attacks is of acute importance. The exchange of alert data between administrative domains can effectively supplement the knowledge gained from local sensors. Intrusion detection systems (IDS's) are typically constrained within one administrative domain. In such an environment, information about the global state of network attack patterns is necessarily unexamined. Global information can aid organizations in ranking and addressing threats that they do perceive and alerting organizations to threats they would not otherwise have recognized.

Most current IDS's do not adequately address the need for information exchange. In addition, the fail–open nature of IDS's makes them a natural target for attacks [1]. Any distributed IDS must enforce mechanisms that support the reliability of its nodes as well as the integrity and confidentiality of the alerts exchanged between those nodes. We propose a collaborative distributed approach to intrusion detection and present a system that addresses the challenges of information sharing.

M. E. Locasto: Columbia University, New York, NY.
J. J. Parekh: Columbia University, New York, NY.
A. D. Keromytis: Columbia University, New York, NY
S. J. Stolfo: Columbia University, New York, NY

**Challenges for Collaboration** Several problems must be addressed before intrusion alert information can be safely distributed among cooperating sites. First, the currently observable rates of alert generation preclude the use of solutions incorporating a centralized system to aggregate and correlate alert information. Administrators need a reasonable chance to respond to the threats that are perceived by the system; replicating all alert information and forwarding it to a common central node places considerable trust in that node and risks both a single point of failure and the increasing congestion of the network near that node. Third, exchanging alert data in a full mesh quadratically increases bandwidth requirements. Fourth, information exchange between administrative domains needs to be carefully managed, as proprietary or confidential network data may escape the boundary of each domain. Finally, any partitioning of the alert information into distinct sets defined by the different domains may cause information loss by disassociating data that should have been considered in the same context.

## II. Related Work

Previous work on distributed intrusion detection has focused mainly on the exchange of data within a single organization. Much of this work has focused on distributed collection and centralized correlation [2]. The DShield project [3] is an example of a centralized repository that receives intrusion alerts from many distributed sources. Only recently has research been performed on systems that support the exchange of data between administrative domains. Cuppens and Miege [4], [5] discuss methods for cooperatively correlating alerts from different types of intrusion detection systems. DOMINO [6] is an overlay network that utilizes the Chord [7] protocol to distribute alert information based on a hash of the source IP address.

CARDS is a prototype distributed intrusion detection system that uses "attack trees", or pre-defined sequences of attack steps. CARDS decomposes global representations of distributed attacks into smaller units (called *detection tasks*) that correspond to the distributed events indicating the attacks, and then executes and coordinates the detection tasks in the places where the corresponding events are observed. While fast algorithms for signature and string matching exist, the best known are of complexity $O(n\ log^3$

ISBN 0-7803-9814-9/$10.00 

*n)*. One notable follow–up work is constructing new attack sequences [8] to keep the signature database up to date.

Krugel *et al.* [9] propose a peer-to-peer system that recognizes attacks in a distributed manner. The system is interesting because their results indicate two things. First, only a relatively small number of messages (seldom more than two in their experiments) need to be exchanged to determine an attack is in progress. Second, their work confirms that a peer–to–peer approach to decentralized intrusion detection is feasible and appropriate. However, the system requires that attack signatures be known *a priori*.

The research closest to ours in aim and scope is the recent work by Lincoln, Porras, and Shmatikov [10]. The authors present a scheme for Internet–scale collaborative analysis of information security threats and claim that it provides strong privacy guarantees to contributers of alerts. The authors sanitize and replicate alert data to enable alert correlation between cooperating peers. While the authors are mostly concerned with the scrubbing of alerts, we examine the need for the construction of an efficient distribution mechanism as well as alert privacy.

Huang and Wicks [11] presents an exploration of the relevant features of attacks, concluding that reconstructing attack strategies is the appropriate method for balancing the cost of distributed correlation with the utility of global knowledge.

A recent study by Moore *et al.* [12] defines the resources needed to counter worm propagation. They conclude that a response needs to be mounted in 2-3 minutes and that participation of nearly all major AS's is required to be effective. The authors are pessimistic about the preparedness of the general Internet for preventing and containing worm outbreaks. Their results suggest that both technological and administrative issues must be addressed before any effective defense can be mounted against such Internet–wide threats. While these numbers are specifically derived in the context of quarantining Internet worms, their results show that current and foreseeable threats demand a cooperative and collaborative approach to achieving security.

Secure multi-party computation (SMC), initiated by [13], [14], [15], [16]), is one of the most fundamental achievements of cryptography in the last two decades. Du and Atallah [17] briefly discuss *privacy-preserving intrusion detection* in the course of enumerating practical applications of Secure Multi-Party Computation. The authors of [10] specifically avoid the use of SMC approaches due to their cost.

## III. Architecture

We adopt two mechanisms in order to cope with the difficulties of distributed correlation and the potential volume of data being correlated. First, the construction of Bloom filters by Worminator is employed to protect the confidentiality of the data being exchanged between domains. Second, efficient information exchange is accomplished with a distributed correlation scheduling algorithm. The scheduling algorithm dynamically calculates subsets of correlation peers that should communicate to exchange Bloom filters. Since information is also compacted by the Bloom filter, correlation between peers becomes extremely cost-effective in terms of bandwidth and processing power.

A key motivating factor for organizations to join a collaboration group in performing distributed intrusion detection is that participants can implement fast mitigation strategies against threats they otherwise would not have known about. For example, DOMINO illustrates the advantage of small blacklists (around 40 entries) that retain their efficacy even when data is fairly stale. Other responses include the content filtering strategies proposed by Moore *et al.* [12], prosecution, or military action.

### A. Requirements

From our discussion above, a set of requirements that guides the design of our system has emerged:

1. The exchange of alert information must not leak potentially sensitive data.
2. Large alert rates hide stealthy activity; any reasonable solution must deal with or reduce the effects of these rates.
3. Centralized repositories are single points of failure and likely unable to correlate the burgeoning amount of alerts.
4. Exchanging alerts in a full mesh quadratically increases the complexity of the problem.
5. Any solution that partitions data among nodes risks information loss by disassociating evidence that should be considered in the same context.

We make several assumptions about the environment the system exists in and the alert information the system exchanges. Our assumptions and choices are intended to carefully balance the requirements of data privacy with the need to derive useful information and actionable intelligence from the alert exchange.

The environment and user base for a collaborative distributed intrusion detection system is an important consideration. We envision cohorts of 25 to 100 organizations exchanging information. Such cohorts can be organizations with similar interests, such as universities, financial institutions, military or government networks, energy companies, news organizations, *etc.*

The sheer volume of alert streams is a critical consideration in the design and evaluation of any distributed intrusion detection system. The size of current (and foreseeable) alert streams demands low–cost processing and correlation. Alert streams can threaten to dominate network bandwidth if they are unnecessarily replicated.

Perhaps the most important decision we make is to employ the use of "watchlists," or lists of IP addresses suspected of subversive behavior. The task of the distributed detection system is not to analyze the network or host

events of other domains, but rather to correlate summaries of alerts to identify attackers. Therefore, watchlists encapsulate the appropriate information to exchange.

### B. Preserving Privacy

While IDS alerts themselves could be distributed, there are two substantial disadvantages to doing this: first, organizations may have privacy policies or concerns about sharing detailed IP data, some of which might uncover who they normally communicate with. Second, these alert files grow rapidly given substantial traffic. While parameters may be tweaked to reduce potential noise, a preferable solution would be to encode the relevant information in a compact yet useful manner.

We provide a compact format via the use of Bloom filters. A Bloom filter is a one-way data structure that supports two operations: insertion and verification, *e.g.,* while no data can be extracted after being inserted in the Bloom filter, it is probabilistically possible to see if specific data has been inserted if presented a second time to the Bloom filter. This is accomplished by creating a compact bit vector (typically between $2^{15}$ – $2^{20}$ entries). Entries are indexed by the hash of the original data, *i.e.,* a high-quality hash of original data (in this case, IPs and port information) is generated, broken up into parts, and these parts are used as indices into the bit vector. Each resolved index in the bit vector is set to 1. This process is typically repeated multiple times (for different parts of the hash and/or different hashes), thereby increasing resiliency to noise or data saturation. Verification is similar to insertion; instead of actually setting bits, the bit vector is examined to determine if the bits are already set. Therefore, the IDS parses its alert output and generates Bloom filters corresponding to (for example) IP/port endpoint data.

Since Bloom filters are compact one-way data structures, we get three benefits:

- *Compactness:* A Bloom filter smaller than 10k bits in size is still able to accurately verify tens of thousands of entries.
- *Resiliency,* even when the Bloom filter is decreased in size: When the Bloom filter is saturated, it starts giving false positives (*i.e.,* multiple data entries resolve to the same locations in the bit vector), but never gives false negatives. The false positives can be ameliorated by tuning or by correlation against multiple alert lists.
- *Security:* By utilizing a one-way data structure, organizations can correlate watchlists without releasing actual IP data, satisfying privacy needs while being able to participate. If further security from outside observers is needed, the dissemination protocol can be encapsulated in a secure tunnel, like SSL, thereby only granting Bloom filter access to the set of participants in the alert list correlation.

### C. Distributed Correlation

A distributed correlation function must overcome the problems of a centralized model while balancing the information loss inherent in partitioning alert data among different nodes. The most straightforward way to accomplish distribution (forwarding all data from every node to every other node) involves a quadratic increase in the amount of data exchanged.

More sophisticated approaches are based on two different theories. The first approach, which mirrors traditional DHT-based P2P networks, creates an explicit mapping from alert data (specifically, source and target IP addresses) to particular correlation nodes. The reasoning behind this approach is that source and target IP addresses are the two most important features (besides target port) of alert data. With data about various machines collected in one node, the majority of correlation can be accomplished at that node without communicating with other nodes (except to distribute results of global interest).

The shortcoming of this approach is that nodes become special cases of the centralized model: they are single points of failure for information pertaining to the IP address range being hashed. In addition, participants in the system may be uncomfortable with storing their raw alert information at a single node. This approach invests too much trust in each node. While a self–healing approach like Chord [7] can ameliorate the loss of a node, previous information stored at that node is at best lost temporarily (for example, in the case of a denial of service,) or corrupted (in the case of the node being compromised). Some of these shortcomings can be mitigated by replicating the data to some number of other nodes; however, it is not clear what the appropriate balance is between fault–tolerance through replication and utilization of network bandwidth and storage space. If data is replicated to every other node, we see an unacceptable quadratic increase in the cost of the system. Furthermore, while DHT-based overlays provide a fast *lookup()* operation, the performance of such networks under churn (rapid series of *join()* and *leave()* operations) is questionable [18].

The second theory attempts to address the limitations of the first approach by introducing a dynamic mapping between nodes and content. This *dynamic overlay network* (as opposed to the largely static mappings of traditional DHT-based overlay networks) implicitly incorporates the notion of churn and does not need to spend time rebuilding neighbor (finger) tables. We observe that a theoretical optimal schedule exists for communicating information. If an oracle existed in the network that answered with the appropriate subset of nodes that should talk given a particular alert, links could be established between these nodes without talking to nodes with irrelevant data (*e.g.*, without a *lookup()* operation).

In this model, we assume that there is a set of nodes $S$ of size $N$. We assume that there is some reliable mech-

anism for any subset of nodes to communicate with each other. There is some discrete unit of knowledge $K$, that if known would provide evidence of a distributed scan. During a distributed scan, some subset of $S$ is scanned and now contains a piece of knowledge $K_i$.

Normally, this knowledge would be discarded as insignificant. However, the optimal schedule allows this set of nodes to perfectly guess which of its neighbors also contains a piece of $K$. Note that each node could also find this information out via the $O(N^2)$ full mesh method of asking each other node in the network. However, we assert that this method is too costly in terms of trust, network bandwidth, and disk storage.

The key idea in this optimal schedule is that the correct subsets of nodes are always communicating. In order to mimic this behavior, the (approximately) correct nodes must talk to each other at (approximately) the right time. One way of accomplishing such a schedule is to pick relationships at random. Another way is to employ a publish-subscribe scheme.

### D. Whirlpool: Network Scheduling

There is a clear need for efficient alert correlation in large–scale distributed networks. To address this need, we introduce the notion of *network scheduling:* the controllable formation and dissolution of relationships between nodes and groups of nodes in a network. These relationships can be envisioned as a dynamic overlay. Our network scheduling mechanism is a procedure for coordinating the exchange of information between the members of a correlation group. The mechanism is controlled by a dynamic and parameterizable correlation schedule.

Our approach is predicated on the previously described theoretical model of the optimal correlation schedule, the shortcomings of a fully interconnected mesh of correlation nodes, and the limitations of the ideal centralized approach to large–scale correlation. The main difficulty is that nodes would most likely discard data that in truth belong to a distributed alert. We must develop a mechanism whereby a node can quickly conference with other peers and determine whether or not a local alert is noise or signifies part of a distributed alert.

The basic architecture of network scheduling is a set of dynamic federations. Nodes that join and leave these federations at various rates. The variance in rates is intended to allow federation groups to retain some stability while expediting the import of new information into the group. Furthermore, this mechanism can be augmented with a distributed learning algorithm that assists in promoting alerts discovered by the distributed correlation.

## IV. Implementation

### A. Worminator

The Worminator platform supports compact watchlist correlation via the replication and use of Bloom filters [19] and uses the Antura network intrusion detection system [20], [21] to generate alerts. The Antura NIDS has been demonstrated to be an order-of-magnitude better at detecting long-term stealthy scans than competing products.

Our initial proof-of-concept version of Worminator ran at specific intervals on the computer running the Antura sensor, parsed its alert output, and generated Bloom filters corresponding to IP/port endpoint data. These alerts were then transmitted to a centralized node using HTTP, and correlation was manually initiated after-the-fact by downloading these Bloom filters from the centralized node. This proof-of-concept prototype demonstrated the feasibility of the idea: in preliminary tests that correlated results from installations of Worminator at two academic sites, three common sources of stealthy surveillance were detected: one each in Beijing, the Phillippines, and a small western US community college. These sources are most probably interested in the test sites to discover weakly protected (but usually more powerful and better connected) university or research machines.

We have since evolved this version to better handle communication latency and privacy requirements. The current version of Worminator is completely pluggable, and supports different sensor and alert types, correlators, and communication frameworks. Just as importantly, it supports *continuous validation*; alerts from the sensor are exchanged immediately, and correlation runs real-time to glean data as soon as possible to help prepare defenses against pending attacks or fast-moving worms.

The goal is to enable sites to maintain a secure *watchlist* of alerts seen locally and from other sites, and to generate a *warnlist* of significant threats if they have been correlated as having been seen at multiple sites. This warnlist can then be reported to network administrators or could be directly mapped to firewall rules to prevent impending attacks. Depending on privacy policies, these local warnlists may also be explicitly replicated to other sites to enable a fast global-scale response.

Worminator consists of approximately 9,500 lines of Java code, and leverages a number of J2EE (Java 2 Enterprise Edition) providers, including the PostgreSQL JDBC provider (for querying databases), the Tomcat JSP/Servlet container (for the user interface), and the JBossMQ JMS provider (for event transport).

## V. Results

### A. Worminator collaborative site experiment

The new version of Worminator was deployed at four different sites in the Northeast: two at Columbia (one on the

perimeter of the Computer Science network and the other on the perimeter of a campus dorm), one at a company in midtown Manhattan, and the last a research institution in Washington, D.C. The Antura sensor was used as the source of alerts at each site, and Java Message Service, a publish-subscribe communication infrastructure, was leveraged to support event distribution amongst the sites. This configuration was run for approximately 96 hours. Information on source IPs and scan times were exchanged.

During the time, approximately 550,000 alerts were generated and exchanged. By far the most of these alerts (roughly 75%) were at the Computer Science network. In addition to traffic passing through the perimeter of the CS network, the CS sensor sees internal traffic between CS machines. (Ordinarily, the dorm network might be more saturated with malicious traffic, but this data was taken during Spring Break, when many undergraduates leave campus. The dorm sensor was also deployed later than the other sites, and had run for approximately two days.)

| Source | # alerts | Alerts/hr | # src IPs |
|---|---|---|---|
| CS | 420425 | 4379 | 5226 |
| Dorm | 9838 | 204 | 290 |
| Midtown | 11394 | 118 | 1831 |
| Wash DC | 116560 | 1214 | 22568 |

Fig. 1. Statistics on the *watchlist*: Sites, number of alerts exchanged, and number of source IPs detected.

As Figure 1 implies, IPs generated more than one alert in some instances; nevertheless, the number of IPs is very large for each site, making individual alerting difficult. A total of 29,731 unique IPs were seen. Next, we ran queries to determine which sources were seen at two, three, or four sites.

| # of sites | # common src IPs | % reduction |
|---|---|---|
| Two | 170 | 99.5% |
| Three | 18 | 99.93% |
| Four | 1 | 99.996% |

Fig. 2. Statistics on the *warnlist*: the number of source IPs detected at two, three, and at four collaborating sites.

The reduction by examining the set of common sources is remarkable – only 18 of the 29,731 original source IPs were observed at three sites. While this does not necessarily discount the other 29,713 IPs, the likelihood that legitimate traffic exists between three of these four unrelated sites is extremely low, and simple port analysis can help confirm this hypothesis. Note that, due to reduced activity at the dorm site, conclusions about the number of common sources at all four sites is preliminary at best, but the one site that matched originated from China. Further analysis from the CS sensor revealed that the machine was probing destination ports 1026 and 1027 – used by the Windows Messenger service [22], strongly suggesting this IP was doing wholesale "pop-up" Internet spamming. While its activity *may* have been benign, such a source is clearly indicative of undesirable traffic, and leveraging the warnlist makes it easy to block such sources, be it undesirable or outright malicious.

We also performed some preliminary analysis on the two-site data to determine if a significant geographic distribution existed (e.g., if more scans originated from any particular site). Figure 3 illustrates the results of the top eight countries (which comprise 87% of the total scans – every other country contributed two or less IPs to the overall total).

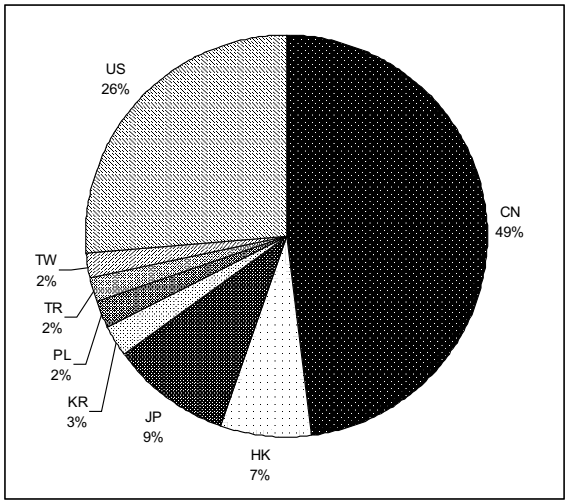


Fig. 3. Geographic distribution of attacks by country for 2-site-detected scans.

As the chart shows, China dominates, with nearly 49% of the top eight countries and 41% overall amongst all countries. This suggests that, despite the "great firewall of China", outbound scans and probes are unaffected and continue to propagate to a broad cross-section of the Internet.

One interesting footnote: while performing analysis on the two-site data, two alerts were generated for the IP `128.9.168.45`. Reverse DNS revealed the URL to be `http://ptr.isi.edu`, which turns out to be an Internet mapping server performing "low-volume" scans. Indeed, this source comprised only .00007% of the total alert exchange, yet we were able to focus on the source with a minimum of effort.

This analysis only scratches the tip of the iceberg; it becomes clear, however, that useful data can easily be gleaned with just a few sites exchanging alerts. We plan to increase the number of participating sites, which will greatly increase the depth and breadth of the types of sources of surveillance and we expect it to further validate our approach.

ISBN 0-7803-9814-9/$10.00  

### B. Alert Rates

Intrusion detection systems face the very real threat of information loss from the sheer rate of available information. Schaelicke *et al.* [23] are decidedly pessimistic about the ability of relatively powerful commodity hardware and network links to absorb peak alert loads, noting that an IDS is effectively neutralized by the loss of alert data resulting from a database unable to keep up with incoming network data. This problem is compounded if multiple NIDS sensors report to the same database system. DShield reports about 10 million alert records added daily. Figure 4 shows the increase in contributed data per month between January 2002 and May 2003.

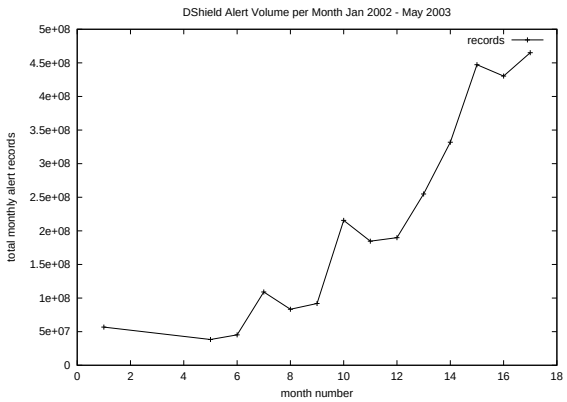


Fig. 4. DShield monthly alert record contributions. The graph is not cumulative, but rather shows the rapid increase in contributed alert information per month as DShield grew in popularity.

Event reduction and aggregation is a critical part of our system. Since we construct watchlists from very little information (an IP address and a port), we are interested in ways of combining different alert information. Reduction can be accomplished by filtering events either at the source sensor or prior to correlation processing. As a trivial example of the latter, consider a series of 100 basic IDS alerts with the same source and destination IP information and alert type information (*e.g., "Host X probed host Y on port Z"*). These alerts can be reduced to a single alert and a frequency. If a significant portion of alert streams are amenable to this type of reduction, we can either perform more expensive processing on the resulting stream, or produce actionable intelligence more rapidly. Inexpensive reduction strategies (like logically grouping attacking IP source addresses in the same /24 subnet) can result in substantial compression, as is aggregation of multiple scan alerts (one per port) by a single source into one overall alert announcing a scan. For an example of the successful application of these reduction strategies,

### C. Network Schedule Evaluation

To evaluate the effectiveness of the Whirlpool network scheduling, we compare it against a full mesh distribution scheme and a random selection distribution scheme. To that end, we introduce a *Bandwidth Effective Utilization Metric* (BEUM). The BEUM is defined as:

$$BEUM = \frac{1}{t * B}$$

where $t$ is the average number of time units it takes the distribution scheme to detect an attack and $B$ is the amount of bandwidth used by the distribution mechanism during that period. B is defined in terms of the total number of nodes, $N$, communicating via the distribution mechanism. Thus, for a full mesh scheme we have:

$$B = N * (N - 1)$$

and the time to discover an attack is $t = 1$. The BEUM for a full mesh distribution strategy is therefore $\frac{1}{N*(N-1)}$. For a system of 100 nodes, the BEUM for a full mesh is $\frac{1}{9900}$

The BEUM for a particular schedule where groups are kept at roughly $\sqrt{N}$ is different and based on the calculation of the bandwidth consumed, $B$:

$$BEUM = \frac{1}{t * B}$$

$$B = N * \sqrt{N}$$

In general, if $t \leq \sqrt{N}$, this particular schedule wins. Specifically, for a system of 100 nodes, the BEUM is $\frac{1}{1000t}$. If $t \leq 9$, this schedule is a better choice than a full mesh. Many other schedules are possible to balance the tradeoffs between bandwidth, coverage, and latency and we are exploring methods for identifying optimal schedules given a set of constraints.

We simulated a randomized scheduling strategy for a system of 100 nodes (performed over 1000 trials). Our simulations indicate that on average, it takes 6 time units before an attack is detected using a repeated random schedule. This time unit requirement satisfies the requirement for $t \leq 10$ we derived for the BEUM. Figure 5 shows that even though some pathological outliers exist, the vast majority of attacks are detected in a relatively short time.

## VI. Conclusions

Distributed intrusion detection is a powerful tool against sophisticated adversaries that do not generate high–visibility events that can be easily caught by traditional means. In this paper, we seek to overcome two challenges: the disclosure of potentially sensitive information to collaborating entities, and the tradeoff between latency and bandwidth used for exchanging information.

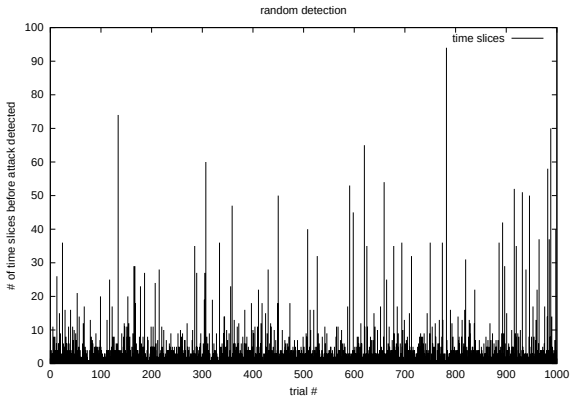


Fig. 5. Number of time slices until random distribution detects an attack. The average number for this particular data plot is 6 time slices.

Worminator creates compact watchlists of IP addresses encoded in Bloom Filters. The use of Bloom Filters minimizes the amount of sensitive information exposed to third parties (that are participating in the distributed IDS), without hampering the detection effort itself. Furthermore, this compact representation lessens the impact of the distributed computation on the network and simplifies the correlation process itself.

One of the key contributions of this paper is the introduction of the notion of a *dynamic overlay* that follows a particular schedule to avoid the problems associated with churn in P2P networks. The system creates federations of nodes that exchange the watchlist and varys the rate at which nodes join and leave these groups. Our experiments and simulations show that even a randomized schedule can detect distributed attacks within an average of six exchange periods without using an unacceptable amount of bandwidth to distribute alert information to peers.

We will continue to study methods for low cost alert distribution and propagation. In addition, we envision developing metrics for ranking the threat level for particular "verticals", or groups of organizations. The deployment of collaborate distributed intrusion detection allows and organization to take mitigation and preemptive threat responses without having been directly attacked.

## References

[1] C. Shannon and D. Moore, "The Spread of the Witty Worm," tech. rep., CAIDA Security Analysis, March 2004. http://www.caida.org/analysis/security/witty/.

[2] S. Stolfo, "Worm and Attack Early Warning: Piercing Stealthy Reconnaissance," *IEEE Privacy and Security*, May/June 2004.

[3] J. Ullrich, "Dshield home page." http://www.dshield.org/, 2004.

[4] F. Cuppens and A. Miege, "Alert Correlation in a Cooperative Intrusion Detection Framework," in *IEEE Security and Privacy*, 2002.

[5] F. Cuppens and R. Ortalo, "Lambda: A language to model a database for detection of attacks," in *Proceedings of the Third International Workshop on the Recent Advances in Intrusion Detection (RAID 2000)*, (Toulouse, France), October 2000.

[6] V. Yegneswaran, P. Barford, and S. Jha, "Global Intrusion Detection in the DOMINO Overlay System," in *ISOC Symposium on Network and Distributed Systems Security*, February 2004.

[7] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-To-Peer Lookup Service for Internet Application," in *Proceedings of ACM SIGCOMM*, August 2001.

[8] P. Ning, Y. Cui, and D. Reeves, "Constructing Attack Scenarios Through Correlation of Intrusion Alerts," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 245–254, November 2002.

[9] C. Krugel, T. Toth, and C. Kerer, "Decentralized event correlation for intrusion detection," in *International Conference on Information Security and Cryptology (ICISC)*, December 2001.

[10] P. Lincoln, P. Porras, and V. Shmatikov, "Privacy–Preserving Sharing and Correlation of Security Alerts," in *Proceedings of USENIX Security Conference*, August 2004.

[11] M.-Y. Huang and T. M. Wicks, "A large-scale distributed intrusion detection framework based on attack strategy analysis," in *Proceedings Recent Advances in Intrusion Detection (RAID)*, 1998.

[12] D. Moore, C. Shannon, G. Voelker, and S. Savage, "Internet Quarantine: Requirements for Containing Self–Propagating Code," in *INFOCOM 2003*, 2003.

[13] A. C. Yao, "Theory and application of trapdoor functions," in *Proc. of the 23th Annu. IEEE Symp. on Foundations of Computer Science*, pp. 80–91, 1982.

[14] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proc. of the 19th Annu. ACM Symp. on the Theory of Computing*, pp. 218–229, 1987.

[15] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for noncryptographic fault-tolerant distributed computations," in *Proc. of the 20th Annu. ACM Symp. on the Theory of Computing*, pp. 1–10, 1988.

[16] D. Chaum, C. Crépeau, and I. Damgård, "Multiparty unconditionally secure protocols," in *Proc. of the 20th Annu. ACM Symp. on the Theory of Computing*, pp. 11–19, 1988.

[17] W. Du and M. J. Atallah, "Secure multi-party computation problems and their applications: A review and open problems," in *Proceedings New Security Paradigms Workshop (NSPW)*, (Cloudcroft, NM), 2001.

[18] J. Li, J. Stribling, T. M. Gil, R. Morris, and F. Kaashoek, "Comparing the Performance of Distributed Hashtables Under Churn," in *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS)*, February 2004.

[19] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, pp. 422–426, July 1970.

[20] S. Detection, "System Detection Releases Antura Vision." http://www.sysd.com/, September 2003.

[21] S. Robertson, E. Siegel, M. Miller, and S. Stolfo, "Surveillance Detection in High Bandwidth Environments," in *2003 DARPA DISCEX III Conference*, April 2003.

[22] LURHQ Threat Intelligence Group, "Windows Messenger Popup Spam on UDP Port 1026." http://www.lurhq.com/popup_spam.html, 2003.

[23] L. Schaelicke, M. R. Geiger, and C. J. Freeland, "Improving the Database Logging Performance of the Snort Network Intrusion Detection Sensor," Tech. Rep. Technical Report 03-10, 2002.